1. Importing necessasary libraries

In [44]:

```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
import warnings
warnings.filterwarnings('ignore')
```

2. Importing data

In [2]:

```
delivery_data = pd.read_csv('delivery_time.csv')
delivery_data
```

Out[2]:

	Delivery Time	Sorting Time
0	21.00	10
1	13.50	4
2	19.75	6
3	24.00	9
4	29.00	10
5	15.35	6
6	19.00	7
7	9.50	3
8	17.90	10
9	18.75	9
10	19.83	8
11	10.75	4
12	16.68	7
13	11.50	3
14	12.03	3
15	14.88	4
16	13.75	6
17	18.11	7
18	8.00	2
19	17.83	7
20	21.50	5

In [3]:

```
delivery_data.head()
```

Out[3]:

	Delivery Time	Sorting Time
0	21.00	10
1	13.50	4
2	19.75	6
3	24.00	9
4	29.00	10

3. Data Understading

In [4]:

```
delivery_data.shape
```

Out[4]:

```
(21, 2)
```

In [5]:

```
delivery_data.isna().sum()
```

Out[5]:

```
Delivery Time    0
Sorting Time     0
dtype: int64
```

In [6]:

```
delivery_data.dtypes
```

Out[6]:

```
Delivery Time    float64
Sorting Time       int64
dtype: object
```

In [7]:

```
delivery_data.describe(include='all',)
```

Out[7]:

```
       Delivery Time  Sorting Time
count      21.000000     21.000000
 mean      16.790952      6.190476
  std       5.074901      2.542028
  min       8.000000      2.000000
  25%      13.500000      4.000000
  50%      17.830000      6.000000
  75%      19.750000      8.000000
  max      29.000000     10.000000
```

Renaming Columns

In [8]:

```
delivery_data = delivery_data.rename(columns={"Delivery Time":"delivery_data","Sorting Time
delivery_data
```

Out[8]:

	delivery_data	sorting_data
0	21.00	10
1	13.50	4
2	19.75	6
3	24.00	9
4	29.00	10
5	15.35	6
6	19.00	7
7	9.50	3
8	17.90	10
9	18.75	9
10	19.83	8
11	10.75	4
12	16.68	7
13	11.50	3
14	12.03	3
15	14.88	4
16	13.75	6
17	18.11	7
18	8.00	2
19	17.83	7
20	21.50	5

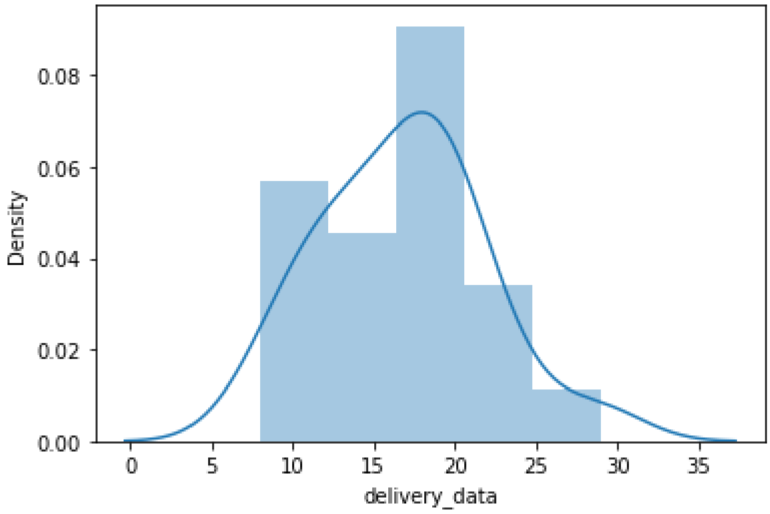
In [45]:

```
delivery_data.info()
sns.distplot(delivery_data['delivery_data'])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   delivery_data  21 non-null     float64
 1   sorting_data   21 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 464.0 bytes
```

Out[45]:

```
<AxesSubplot:xlabel='delivery_data', ylabel='Density'>
```

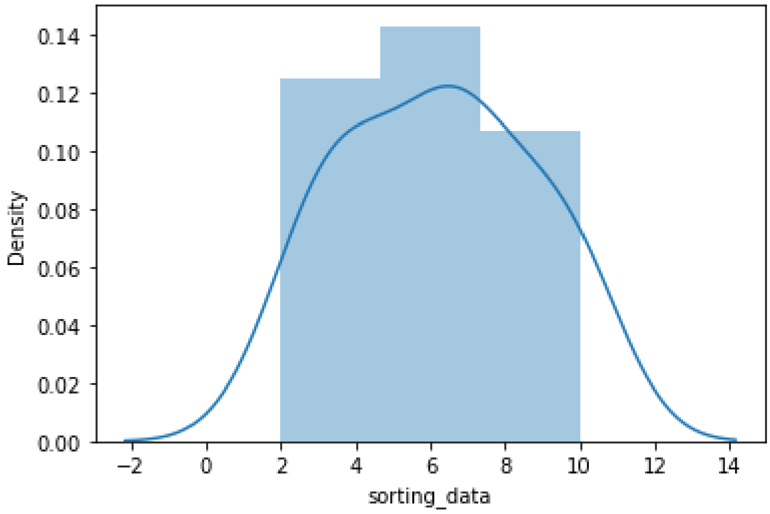


In [46]:

```
sns.distplot(delivery_data['sorting_data'])
```

Out[46]:

```
<AxesSubplot:xlabel='sorting_data', ylabel='Density'>
```



In [9]:

```
delivery_data
```

Out[9]:

	delivery_data	sorting_data
0	21.00	10
1	13.50	4
2	19.75	6
3	24.00	9
4	29.00	10
5	15.35	6
6	19.00	7
7	9.50	3
8	17.90	10
9	18.75	9
10	19.83	8
11	10.75	4
12	16.68	7
13	11.50	3
14	12.03	3
15	14.88	4
16	13.75	6
17	18.11	7
18	8.00	2
19	17.83	7
20	21.50	5

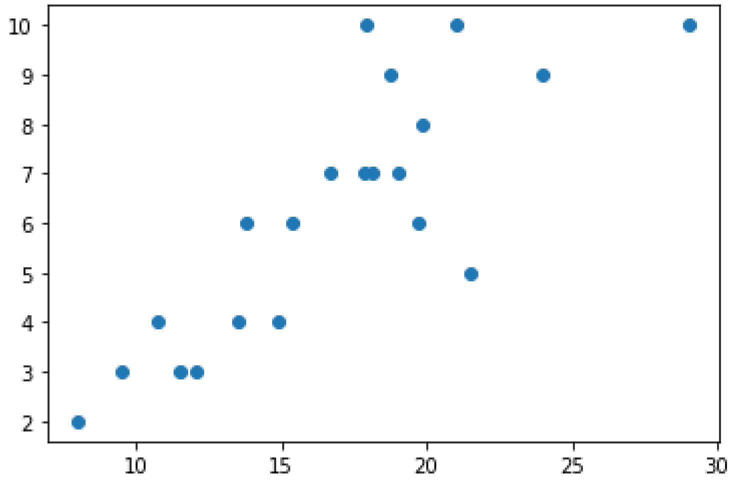
3.[b] Check Assumptions are matching

In [11]:

```
plt.scatter(x = 'delivery_data',y = 'sorting_data',data=delivery_data)
```

Out[11]:

```
<matplotlib.collections.PathCollection at 0x1ec49eaa970>
```



In [12]:

```
# correlation analysis
delivery_data.corr()
```

Out[12]:

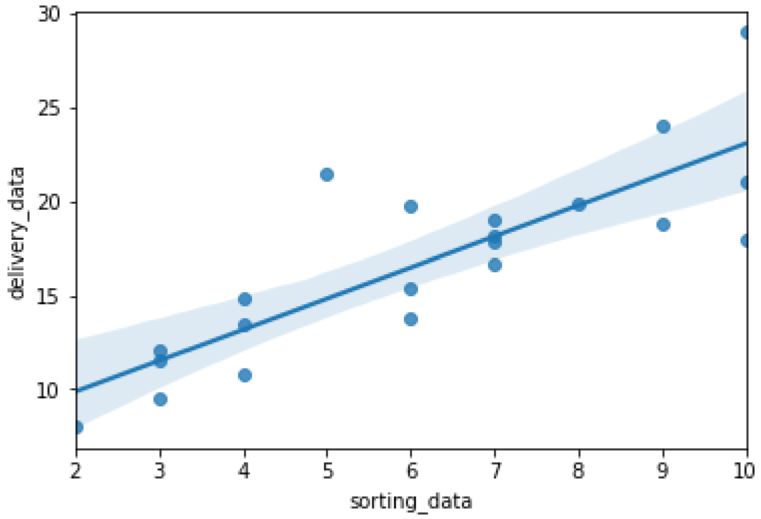
	delivery_data	sorting_data
delivery_data	1.000000	0.825997
sorting_data	0.825997	1.000000

In [13]:

```
sns.regplot( x='sorting_data', y='delivery_data', data=delivery_data,)
```

Out[13]:

```
<AxesSubplot:xlabel='sorting_data', ylabel='delivery_data'>
```



4. Model Building || Model Training

There are basically 2 libraries that support Leniar Regression algorithm

1. Statsmodels libraries

2. sklearn libraries

In [36]:

```
import statsmodels.formula.api as smf
```

In [26]:

```
# Odinary Least square
linear_model=smf.ols(formula = 'delivery_data~sorting_data', data = delivery_data).fit() #
linear_model
```

Out[26]:

```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x1ec4bc01b
b0>
```

6. Model Testing

In [31]:

```
# finding Coefficient parameters
linear_model.params
```

Out[31]:

```
Intercept       6.582734
sorting_data    1.649020
dtype: float64
```

In [32]:

```
# Finding tvalues and pvalues
linear_model.tvalues, linear_model.pvalues
```

Out[32]:

```
(Intercept       3.823349
 sorting_data    6.387447
 dtype: float64,
 Intercept       0.001147
 sorting_data    0.000004
 dtype: float64)
```

In [29]:

```
# Finding Rsquared Values
lin_model.rsquared, lin_model.rsquared_adj
```

Out[29]:

```
(0.6822714748417231, 0.6655489208860244)
```

7. Model prediction

Manual prediction for say sorting time 5

In [34]:

```
delivery_time = (6.582734) + ( 1.649020)*(5)
delivery_time
```

Out[34]:

```
14.827834
```

8. Automatic prediction for say sorting time 5, 8

In [38]:

```
new_data = pd.Series([5,8])
new_data
```

Out[38]:

```
0    5
1    8
dtype: int64
```

In [40]:

```
data_pred = pd.DataFrame(new_data,columns = ['sorting_data'])
data_pred
```

Out[40]:

	sorting_data
0	5
1	8

In [41]:

```
linear_model.predict(data_pred)
```

Out[41]:

```
0    14.827833
1    19.774893
dtype: float64
```

In [48]:

```

#Thanks Assignment Completed Delivery_time
#Question :- Predict delivery time using sorting time
#Manjunath Pujer 6th Nov 2021
```

In [ ]: